CS335A: Milestone 1

Sawan H N (210952) Shrilakshmi S K (211012) Yashas D (211199)

March 3, 2024

1 Required Environment

For the environment setup, it is essential to have Flex and Bison installed. On Linux systems, these tools can be installed using the commands `sudo apt install flex` for Flex and `sudo apt install bison` for Bison, respectively. Additionally, the system must have the `g++` compiler available for use. To visualize the `.dot` file produced when the parser file is executed, the Graphviz tool is utilized.

2 Instructions to run the code:

Go to directory `cs335-project-39` and run the following commands in terminal:

```
cd milestone/src
make clean
make
cd ..
./run.sh -i <input.py> -o <output.pdf>
```

All the tests are in `tests` directory and all the outputs are in `outputs` directory. Note that `make` might fail sometimes on Windows. Please retry if it does. On Linux, if any file denies permission, please execute `sudo chmod +x <filename>` for that file.

3 Options

How to use: `./run.sh` [options]
Options:
`-i Specify input (.py) file`
`-o Specify output (.pdf) file`
`-v Enable verbose mode`
`-h Show help`

4 Lexical Analyzer

We have written rules for all the tokens necessary to implement Python 3.8 grammar. The following list encompasses key implementation details of lexer.

4.1 Implicit Line Joining

To enhance the width of cases covered and facilitate certain syntax structures, our system supports implicit line joining under all conditions. Some key conditions include:

- Nested function calls, where one function call is used as an argument within another function call.
- Function calls placed within lists.
- The declaration and use of multi-dimensional lists.

To manage this functionality, we employ a state-based approach, transitioning to a distinct state upon encountering an opening bracket (`(` or `[`) and reverting to the previous state when a closing bracket (`)` or `]`) is encountered. This method allows us to differentiate between parentheses and square brackets, ensuring support for the scenarios outlined above.

For state management, we utilize a stack that records state transitions, enabling the system to return accurately to prior states once bracket pairs are closed. This stack mechanism tracks the depth and context of nested structures.

4.2 Indentation

We use a separate stack dedicated to managing indentation levels. This stack helps us generate the appropriate indent and dedent tokens, crucial for understanding the structure of the code, especially in languages where indentation is syntactically significant. A special state is designated for handling situations where multiple dedent tokens need to be passed to the parser, ensuring the parser receives the correct cues to interpret the code structure effectively.

4.3 Other Cases

We have also handled explicit line joining, blank lines and comments especially within multiline lists.

5 Changes to Grammar

We have implemented Python 3.8 grammar for this milestone. To remove shift-reduce conflicts, reduce the number of states and make the grammar more comprehensible, we have further made the following changes.

- Included operators and operands in same production rule for expressions.
- Removed epsilon productions causing shift-reduce conflicts.
- Removed optional semicolon and comma at the end of a sequence of small statements and list of arguments or tests respectively. This has been done because official python shows warning when semicolon is placed at the end of sequence of small statements. This change does not affect future stages of compilation in any manner.
- Removed right recursion in some productions to reduce the number of non-terminals.
- Introduced few non-terminals to incorporate the regular expressions given in standard grammar to Bison.

6 Constructing AST

We first built the parse tree and further made the following changes to construct an AST.

- Removed all non-terminals with single child from the graph. We connected the children of child of non-terminal with single child directly to the said non-terminal. This vastly improves comprehensibility of AST.
- For expressions, the operators are made parents and the operands are its children.
- Labelled the edges with relevant attributes. For example, the type of variable such as `int` or `str` or `bool` has an incoming edge with the label `"type"`. Another example is in if-statements, the condition expression has an incoming edge with label `"if"` and the body of if condition has an incoming edge with label `"then"`. To improve comprehensibility, the enveloping brackets such as `()` and `[]` are also labeled on edges. Similarly, edges are labelled in function definitions, class definitions, for statements and while statements.
- As a result of labelling edges, we have removed some redundant terminals from AST such as `IF, ELSE, ELIF, FOR, IN, WHILE, COMMA, SEMICOLON, COLON, DEF, CLASS, OPEN_SQUARE_BRACKET, CLOSE_SQUARE_BRACKET, OPEN_ROUND_BRACKET` and `CLOSE_ROUND_BRACKET`. These terminals are easily understood from the edge labels.

7 Additional Features

We have handled all the features as required by milestone. Additionally we have supported the following features.

- Multi-dimensional lists
- Comments under all cases including in-between multi-line lists, multi-line function calls and definitions.
- Explicit type casting
- Escape sequences in strings

- Assert statements
- Any number of spaces are supported for indentation (similar to official Python). It need not necessarily be 2 spaces.

Some features like nonlocal statements, import statements, del keyword, with statements, pass keyword and raising exceptions can be easily included in the parser. But these are excluded due to uncertainty in implementation of these features in future stages of compilation.

8 Reporting Errors

The lexer reports indentation error and invalid character error with line number and exits the program. We have used `%define parser.error verbose` in parser to report the errors where the input does not obey the grammar's productions. It also reports expected token and line number where error is encountered.